Malcolm Smith

Nov 9, 2023 | **CV Supplement**

Recommendations and Endorsements

Please don't just take my word. As Data Engineers we like empirical analysis of qualitative and quantitative data. I invite you to view the recommendations and endorsements collected during my tenure at Keysight Technologies. I've summarised a few recommendations for your convenience below:

"Malcolm recognises that leadership goes beyond delivering results; it's about cultivating a culture of respect, inclusivity, and empathy."

M. Khanna | **Lead Data Engineer**

"His brilliance is evident not just in his deep technical acumen, but also in his rare ability to demystify complex topics for those less technically inclined."

R. Kardjaliev | **Lead Software Engineer**

"Malcolm truly understands the importance of collaboration between development and operations and demonstrates a fantastic ability to bridge the gap between the two domains [...] to deliver reliable and scalable systems."

A. Rickman | **Platform Automation Lead**

"Malcolm not only excels in data modeling, machine learning, cloud technologies, end-to-end delivery of data workloads in production systems, technical architecture, and building test suites at all levels but also fosters an environment where everyone is encouraged to give their best."

M. Gaye | **Senior Data Engineer**

"Outstanding analytical skills and attention to detail, coupled with excellent communication, ensures he can engage stakeholders across all levels and clearly articulate concepts all the way through to detailed technical implementations in code."

S. Austin | **SVP Engineering**

"He was the go to person for all technical queries in the team. He is very keen on adopting quality practices to ensure appropriate test coverage across various levels of the test pyramid is achieved."

P. Baluvaneralu | **Software Test Lead**

"Malcolm is not only an empathetic leader but also possesses a customer-first mindset. He excels in crafting clear and well-structured requirements, ensuring the smooth execution of our projects."

D. Delgado | **Product Manager**

"On customer calls and in demos, Malcolm has an engaging style and a great skill in simplifying explanations of complex processes."

Phil Turner | **Senior Solutions Engineer**

Responsibilities/Accountabilities Held in the Last 5 Years

Data Engineering | 8 Years
Deliver innovative data solutions to underpin business-critical requirements • Design highly available distributed data processing at scale via batch and stream • Lead technical implementations across Data Lake, Data Warehouse, Data Modeling, ETL design, and more • Support Data Science • Engineer and optimise Machine Learning (and processes) • Drive efficiency, effectiveness, and quality • Technology and vendor selection • Identify, design, and implement improvements to existing data solutions • DataOps • Production support for all issues • Data governance, security, and compliance.

Platform Engineering | 5 Years
Determine the most appropriate route to get a model/database/ETL from prototype to production to deliver business value early • Architect cloud infrastructure with minimal risk, time, and cost for maximal durability, availability, and performance • Support Data Science • Manage platform capacity, scaling, and budgets • AWS Well-architected • Efficient, maintainable, and extensible Infrastructure as Code • Operationally run production systems including debugging and live incident management.

DevOps | 8 Years
Establish and maintain DevOps best practices such as version control, collaboration, automation, monitoring, and compliance at high velocity • Uphold and embody the philosophies of the DevOps ethos throughout continuous release cycles • Support truly self-organising, cross-discipline, and operationally enabled engineering teams • Design and deliver Continuous Integration/Continuous Delivery • Leverage modern GitOps practices for Operational Excellence in platform deployment.

Technical Architecture | 8 Years
Drive (and contribute to) discussions supporting decision making with respect to system infrastructures, architectures, systems, libraries, workflows, processes, etc. • Take strategic approaches to problem solving and see the big picture at all times • Communicate with stakeholders and audiences at different levels • Understand of the importance of operational, security, and legal factors when designing solutions • Articulate trade-offs and make informed decisions around design and technology choices • Understand the maintainability of design choices • Comprehend and manage Technical Debt at all stages of design.

Technical Leadership | 9 Years
Critical decision-making • Thought leadership • Lead by example and encourage software engineering best practices • Mentor and coach less experienced Engineers • Facilitate technical discussions, encourage collaboration, and provide direction amidst uncertainty • Be on-point for customer and operational escalations • Work in a modern, agile development team following best practices like code review, TDD/BDD, CI/CD, and pairing • Support engineering teams in being performant and focussed • Identifying, prioritising, and paying off Technical Debt.

Line Management | 8 Years
Manage a talented and high potential team of cross-discipline Data, Platform, DevOps, and QAOps • Align personal growth to business and personal interests • Establish a culture to be proud of (not just results to be proud of) • Lead by example in DEI • Motivate teams and individuals • Actively manage disruptive and/or poorly performing engineers • Find talent to meet evolving business needs.

Agile SCRUM | 10 Years
Collaborate in project governance, writing requirements/stories, roadmap planning, and release planning • Lead ceremonies in planning, refinement, retrospectives, demos, standups, etc • Product Ownership • SCRUM Mastery • Actively observe, troubleshoot, and manage team velocities.

Software Engineering | 16 Years
Contribute and guide in building reliable, scalable, extensible, maintainable, testable, and deployable software • Deliver web applications in both frontend and backend disciplines • Solidly apply computer science principles, with confidence, across a broad selection of languages, technologies, and patterns • Design and contribute to testing at all levels of the testing pyramid.

Mastery in Data, Platform, and Software Delivery

Ask me a question on any of the following. I bring comprehensive experience in all* enumerated items.

** Except items marked as WIP or Light - here I can speak from theoretical/basic knowledge with a growing practical experience!*

Data Practices

Distributed Data Architecture
Batch/Micro-batch/Stream
Data Modeling and Design
Data Cleaning/Wrangling
Data Quality/DataOps
Feature Generation and Mining
Advanced SQL and BI
Data Enablement and Access

Data Processing

Exactly Once/ALO/AMO
Extensive ETLs (Java & Python)
Hadoop, HDFS, Hive
Map Reduce, Spark
Pandas, NuMPI, OR-Tools (etc.)
Machine Learning (WIP)
MLOps (WIP)

Data Storage

Data Lake/Lakehouse
Data Warehouse/Mart
SQL (Inc. MySQL/PostgreSQL)
NoSQL (Inc. Elastic/Mongo)
Schema Mgmt. (on write and read)
Multi-tenanted Data Isolation
Data Migration/Reprocessing
Separation of Compute and Store

Cloud Platform (AWS)

API Gateway
Kinesis Streams and Firehose
DynamoDB
Relational Database Service
Glue Data Catalog & Jobs
Athena & Workgroups
Elastic Map Reduce (EMR)
Elastic Container Service
Elastic Container Repository
Elastic Kubernetes Service
Lambda and Fargate
Simple Notification Service (SNS)
Simple Queue Service (SQS)
Simple Email Service (SES)
CloudFront/Route53
CloudWatch
Load Balancers
VPC/Subnet/SG (etc)
EC2 Spot/Fleet/Reserved
QuickSight Business Intelligence
Well-architected Framework
Cloud Budgeting/Management

Operational Excellence

DevOps/GitOps
IaaS/PaaS
Terraform/Terragrunt
Puppet
Docker
Kubernetes
Platform Scalability/Redundancy
Fault-tolerance/Failure recovery
Elastic Stack (ELK)
Grafana (WIP)
Jenkins
GitLab Automation

Security/Governance

Data Encryption
ISO27001/SOC-2
Management of PII
OAUTH2/OIDC AuthN
Fine-grained AuthZ
IAM Systems (inc Keycloak)
Single Sign On (SSO)

Software Engineering

Technical Architecture
Software Patterns/Principles
Architecture Patterns/Principles
Microservices
Node, JavaScript, Python
Java, PHP, C# (plus others…)
CLI and Window/Mac/Linux

Testing

TDD and BDD
Testing Pyramid
Cucumber/Gherkin
Pytest/Mocha/JUnit (etc.)
Selenium
Browser Stack/Source Labs
Keysight Test Automation

Team Leadership

Line Management
Performance Management
Recruitment/Dismissal
Personal Development
Mentoring/Coaching

Delivery Leadership

Requirements Writing
Roadmap Planning
Release Planning/Governance
SCRUM-master
Product Owner

Methodology

Agile
SCRUM
Kanban
Lean
Cross-discipline

Case Studies

Ask me a question on any of the following. These case studies serve as tangible exploration of real-world knowledge and skill.

1) Scalable SaaS Cloud Data Platform — Data Architecture and Platform Architecture

Upon taking Team Lead of the Real Customer Insights (RCI) project, the second generation of the SaaS RCI project was commissioned. Each geographic (regional) cloud deployment was to be multi-tenanted. I delivered the end-to-end design and build of a highly scalable, resilient, isolated, and secure data platform that collects user telemetry from software in the wild and catalogs it ready for processing. Exactly which transformations were unknown, as was the mode of data consumption. For this reason flexibility and extensibility ranked highly compared to other competing facets.

Solution: Jenkins, Docker, Terraform, API Gateway, Kinesis, Kubernetes, S3, Glue, Pytest.
Takeaway: Schema management/drift in a schema-on-read Data Lake gave us some trouble, but probably no more trouble than a schema-on-write would have when we didn't have solid data modeling/access requirements. Costs were favourable compared to the equivalent Data Warehouse. Data could be kept cold/very cold with nominal incremental cost - but in truth this was never really leveraged. 3 years later we re-architected Kubernetes as we did not see the expansion we forecasted (case study below).

2) JavaScript Instrumentation and Telemetry — Software Architecture and Test Automation

Data collection required instrumentation of Web Apps. This meant designing, publishing, and distributing a telemetry SDK, plus the means to publish and distribute instrumentations to be deployed directly into the customer's production and pre-production environments. Of note is my modular design for triggering, collecting, and transporting non-PII (Personally Identifiable Information) data securely, performantly, and resiliently to the data platform. 'Drop in' integration was priority one. Next was 'garbage in, garbage out'.

As well as coding defensively for browser internals, network, and cross-browser support, a comprehensive CI/CD automation pipeline was designed and built over time. This included static code analysis, unit/integration testing, rich E2E (UI) cross-browser testing with BrowserStack and later SourceLabs, finally arriving at publishing and distribution via CDN and package managers. This provided on-demand publishing and daily regression.

Solution: Node and JavaScript, Webpack (etc.), ESLint, Mocha, Pytest, Gitlab Automation, S3, CloudFront, BrowserStack, Source Labs.
Takeaway: Despite advocacy, business buy-in for investing in automation/testing never surfaced. This automation grew as a result of dedication above-and-beyond because on-the-ground perspective dictated that this was indeed the "right" path for quality assurance. Not ideal, but together the team arrived at an outcome that we felt was important - delivering high quality without manual handle-turning.

3) User Journey Processing — Data Architecture and Data Engineering

The first major data transformation chunked sequent user events into user sessions. I challenged real-time requirements and these were subsequently thrown out, so batch-processing was implemented via a simple MapReduce job on a schedule.

Solution: Terraform, EventBridge, Lambda, EMR, Java MapReduce, S3, Glue.
Takeaway: MapReduce in an ephemeral EMR cluster ended up being incredibly stable and reliable, but it also ended up being the only Java MR ETL in the domain (Python ETLs came next). With the benefit of hindsight - holding off and evaluating an alternative transformation in PySpark would yield better performance and reduced technical debt but risk a late delivery.

4) Unsupervised Performance Timing Anomaly Detection — Data Science and Data Engineering

Step changes in website performance lead to significant financial ramifications. Using performance timing from telemetry collected, we formulated a Data Strategy with the goal of identifying anomalous timing points and reporting cohorts that were observed as potentially contributing factors. Early discovery led to positive outcomes for RRCF. Due to its lightweight footprint and training requirements, I approved it as an agreeable candidate and proceeded to plumb it into a distributed data mining workload that exploded forecasted vs actual values for a broad range of mined segments.

Solution: Jupyter, Robust Random Cut Forest (RRCF), Docker, EventBridge, Lambda, EMR, ECS Fargate, PySpark, S3, Glue, Pytest, QuickSight.
Takeaway: Due to time pressures, evaluation of alternate models and hyperparameters was cut short. A reasonable period of Data Engineering followed, but a prolonged period of settling was observed, requiring multiple patches to the training sets/routines and the data mining surrounding it. On balance - we got to production earlier, but the overall cost may have been higher. Given more investment additional algo's were proposed to infer more consumable insight and reduce the burden on the human consuming the data.

5) Unsupervised Digital Behaviour Clustering — Data Science and Data Engineering

Customers told us our MVP of user journey modeling for model-based test automation (effectively a sort and limit) was weak. Identified the core problem as cardinality and recognised the need to break journeys down into their constituent parts/structures to make comparison plausible. Through workshops, discovery, and prototyping, we arrived at n-gram structures that allowed for small-degrees of variation tolerance due to itemset mining. With this in-house "Digital Behaviour" definition in hand, I proposed reducing cardinality by labeling, merging, and aggregating to produce a compelling dataset and analytics. Nearest neighbour algos meant we could present some highly explainable and tangible Business Intelligence for customers to see "into" our algorithm (without needing to know anything about it).

Solution: Jupyter, Graph, N-Gram, Itemset Mining, Nearest Neighbour, Terraform, EventBridge, Lambda, EMR, PySpark, S3, Glue, Pytest, QuickSight.
Takeaway: A highly successful discovery phase led to early positive outcomes. The Data Engineering and Operationalisation of the pipeline was non-trivial, but thanks to a strong team working very well together, we established a highly resilient and performant pipeline and has stood the test of time. Of note - higher variance led to higher memory and some initial settling of the workloads/scale/cost management.

6) Digital Behaviour Optimisation — Data Science and Data Engineering

The "Digital Behaviour" dataset was compelling. Aggregation provided us metrics we thought we could leverage but it was still too crude for consumption. By modeling the business domain and the Digital Behaviour data set I arrived at a set of features we could confidently use as goals/constraints/heuristics to filter, minimise, and prioritise the crude data set. The problem domain of combinatory constraint and optimisation was identified, and Google's OR tools is the leading solution. Via workshops I worked with Data Engineers to rapidly iterate a series of integer programs which could effectively represent the Digital Behaviour and the business domain's goals/constraints with very successful outcomes. We arrived at a high confidence in minimal "effort" for maximal "assurance", then layered more algo's to identify the pathways from the reduced search space that return maximal re-use. This created a compelling narrative for published artifacts to be consumed via Business Intelligence.

Solution: Jupyter, OR Tools, Pandas, Docker, Terraform, Lambda, ECS Fargate, S3, Pytest, QuickSight.
Takeaway: Data Engineering and Operationalisation of the pipeline was smooth enough, but testing across the piece was starting to become challenging and a drag factor. We invested in a new Pytest blueprint that could rapidly copy fixtures, trigger workloads, identify workload success/failure scenarios, and deeply interrogate the workload's outputs. The resulting test automation suites paid back with dividends.

7) Re-architecting Kubernetes for ECS Fargate — Platform Architecture and Technical Debt

I responsibly conducted a weekly review of the platform, including operational costs/budgets. Following an upgrade to Terraform and Kubernetes to remain within LTS of the components, I quickly identified massive spikes in cost. Following multiple rapid iterations of triage and cost stemming with Platform and Data Engineers, we reached diminishing returns. In short, the cetralised kubernetes modules had bloated to a massive degree, and were no longer a good "fit" for our project's much simpler use-case.

I was left with a decision; fork the modules and fully own a second entire deployment of K8s or migrate away. Forking was deemed net negative due to the operational and opportunity costs of maintaining k8s in the platform given our simplistic requirements. Taking our containers over to ECS Fargate was deemed the forward-thinking move - freeing up both team capacity and operational budget. Following an MVP of operational excellence (scalability, network resiliency, redundancy, fault-tolerance, failure recovery) the switch was made. This move has been validated by AWS Architects and appears to follow a rising trend of companies experiencing high cost and high skill after day 1.

Solution: Terraform, Docker, ECS Fargate, CloudWatch, Route53.
Takeaway: Operational costs were slashed, and 20% of quarterly team capacity freed up. My advocacy was for improved inter-team requirements mapping/management - as the real problem was the modules growing to fit one use-case without due consideration of others. Delays to paying technical debt exacerbated this.

8) Migrating Jenkins to GitLab Automation — DevOps and Technical Debt

When the RCI project was first commissioned Jenkins was the tool of choice for DevOps in the department. One year later there was a department-wide commitment to GitLab Automation. New services came online under GitLab, but investment was never prioritised to move existing pipelines from Jenkins. Over years, Jenkins was neglected and flagged as a significant risk to team productivity were it to become unhealthy. Fortunately, it didn't. Multiple attempts were made to get the technical debt prioritised, but it only ever made it to 2 or 3 on the list of priorities, so was never prioritised officially. Following above-and-beyond activity and seeded by hackathon projects, I eventually signed off the migration of all services to GitLab Automation.

Solution: Jenkins, Gitlab Automation, Docker, IAM.
Takeaway: In my experience, most technical debt is paid when features/enhancements touch an affected area and estimation increases accordingly. Where technical debt exists without customer-facing features - prioritisation becomes particularly challenging! Objectively balancing productivity, risk, impact, effort, and team motivation is tricky. Getting buy-in and securing investment is trickier still - even in an organisation that has a relatively healthy attitude towards technical debt!

9) Thought Leadership and Writing Papers — Innovation and Stakeholder Management

Over the years I have exhibited constant motion and thought leadership, all of which drove engineering and product functions to strive for more innovative solutions, seek greater quality, and peek beyond perceived limitations. I've been fortunate to work with gifted engineers, sit in customer calls, and pitch to customers and strategic partners. I've networked to become highly connected to those who are the most creative and forward thinking contributors. Inspired by the state of the art and the short-comings of our existing offerings, I have written and collaborated on countless papers attempting to articulate the various problem spaces, solution spaces and academic research available, in order to put forward notional solutions. Some were merely socialised for enablement, but all of them built up a rich tapestry of evolving possibilities.

Solution: The internet, Jupyter, Confluence, Whiteboards, in-person workshops.
Takeaway: Don't be afraid to fail - just do it fast and learn faster. Don't sweat the details. Clearly articulate the problem space. Apply rigor, not obsession. Be honest and objective - recognise your bias. Summarise the related academic research or case studies, where possible.